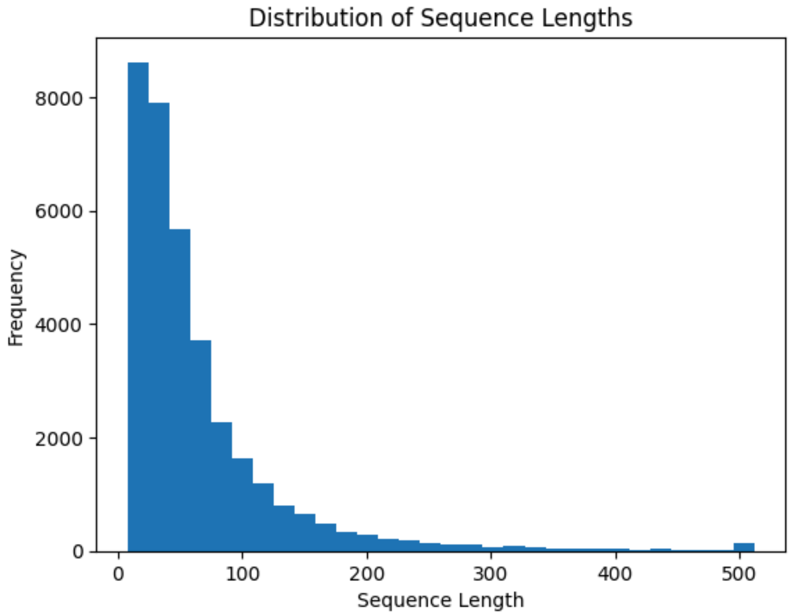
Report of Data Mining Hw2, Review Rating Prediction

109550027, 紀竺均

1. How do you select features for your model input, and what preprocessing did you perform to review text?
    a. For the ratings, I change them to a list with length=5, and set the label's index = 1 in the list. Ex: rating = 2.0 -> [0, 1, 0, 0, 0].
    b. After viewing many training texts, I found that there are multiple
 tags in the text, so I removed them while preprocessing the data.

2. Please describe how you tokenize your data, calculate the distribution of tokenized sequence length of the dataset and explain how you determine the padding size
    a. tokenization: First, I format 'title' and 'review' into a single string per row. Then, I use the BERT tokenizer to convert text into tokens since I use Bert Model later in this assignment. It splits words into smaller units and converts units to their respective token IDs (numerical representations understood by the BERT model.)
    b. distribution of tokenized sequence length: I write a block to calculate the distribution of sequence length. As you can see from the plot below, max sequence length (500+) is an extreme case, most sequence length is around 100~200, so I set the max padding size to 128.

```
tokenizer(texts, truncation=True, padding='max_length',
max_length=128)
```



3. Please compare the impact of using different methods to prepare data for different rating categories

At first, I used the one-hot encoded format mentioned in question 1. However, considering the ratings have an order (1 is worse than 5), I decided to conduct two experiments.

1. Use origin scalar labels.

```
def compute_metrics(eval_pred):
   predictions, labels = eval_pred
   predictions = np.argmax(predictions, axis=1)
   return {
       'accuracy': evaluate.load("accuracy").compute(predictions=predictions,
references=labels),
       'f1': evaluate.load("f1").compute(predictions=predictions,
references=labels, average='weighted'),
   }
```

The strength:

- Simple, no additional preprocessing needed.
- Smaller storing memory, faster training time.
- Represents the order or hierarchy, helping the model leverage the ordinal relationship between classes.

The weakness:

- Limit flexibility in loss function selection

2. Use the one-hot encoded format

```
def compute_metrics(eval_pred):
   predictions, labels = eval_pred
   predictions = 1/(1 + np.exp(-predictions))
   predictions = (predictions > 0.5).astype(int).reshape(-1)
   return {
       'accuracy': evaluate.load("accuracy").compute(predictions=predictions,
references=labels),
       'f1': evaluate.load("f1").compute(predictions=predictions,
references=labels, average='weighted'),
   }
```

The strength:

- Model Flexibility, Each class gets its own error gradient during backpropagation.

The weakness:

- Did not utilization the ordinal relationships (1 lower than 5)
- Increase the model's complexity and is inefficient.